UNCLASSIFIED

AFIT/EN/TR94-06

Air Force Institute of Technology

Exploiting IB Assignments for Approximating Marginal Probabilities

Eugene Santos Jr. Solomon Eyal Shimony

September 28, 1994

Exploiting IB Assignments for Approximating Marginal Probabilities

Eugene Santos Jr.
Dept. of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-7765
esantos@afit.af.mil
Solomon Eyal Shimony
Dept. of Math. and Computer Science
Ben Gurion University of the Negev, 84105 Beer-Sheva, Israel
shimony@bengus.bitnet

April 21, 1994

Keywords: Probabilistic Reasoning, Bayesian Belief Networks, Relevance, Belief Updating, Belief Revision.
Acknowledgements: This research was supported in part by AFOSR Project #940006.

Abstract

Computing marginal probabilities (whether prior or posterior) in Bayesian belief networks is a hard problem. This paper discusses deterministic approximation schemes that work by adding up the probability mass in a small number of value assignments to the network variables. Under certain assumptions, the probability mass in the union of these assignments is sufficient to obtain a good approximation. Such methods are especially useful for highly-connected networks, where the maximum clique size or the cutset size make the standard algorithms intractable.

In considering assignments, it is not necessary to assign values to variables that are independent of (d-separated from) the evidence and query nodes. In many cases, however, there is a finer independence structure not evident from the topology, but dependent on the conditional distributions of the nodes. We note that independence-based (IB) assignments, which were originally proposed as theory of abductive explanations, take advantage of such independence, and thus contain fewer assigned variables.

As a result, the probability mass in each assignment is greater than in the respective complete assignment. Thus, fewer IB assignments are sufficient, and a good approximation can be obtained more efficiently. IB assignments can be used for efficiently approximating posterior node probabilities even in cases which do not obey the rather strict skewness assumptions used in previous research. Two algorithms for finding the high probability IB assignments are suggested: one by doing a best-first heuristic search, and another by special-purpose integer linear programming. Experimental results show that this approach is feasible for highly connected belief networks.

1 INTRODUCTION

Computation of marginal probabilities of variables in a Bayesian belief network is a problem of particular research interest for the probabilistic reasoning community. Although a polynomial-time algorithm for computing the probabilities exists for polytrees [21], the problem was proved to be NP-hard in the general case in [9]. Several categories of exact algorithms exist for computing posterior probabilities: clustering and junction-trees [23, 20], conditioning [10], arc reversal [36], and term evaluation [24]. Many variants of these algorithms attempt various refinements of these schemes, e.g. [13]. All of these algorithms have exponential-time algorithms in the worst case, where the runtime is a function of the topology and the number of states each variable can assume (in this paper we only refer to networks where each variable has a finite number of states).

In the hope of avoiding an exponential runtime, a host of approximation algorithms have emerged. As it turns out, theoretically, even *approximating* marginal probabilities in belief networks is NP-hard, an thus there *is no* polynomial-time (deterministic) approximation algorithm unless P=NP [11]. Most approximation algorithms are less affected by network topology, and are dependent on the actual probabilities as to their runtimes and quality of approximation. If the topology of a given network is such that exact algorithms is expected to take a long runtime, it may be advisable to run an approximation algorithm and hope that the probabilities are such that we can get a good approximation in reasonable time for the problem instance at hand. In addition, most approximation algorithms have an *anytime* behaviour, which facilitates trading off time for precision in a graded manner.

Two major categories of marginal probability approximation algorithms exist: randomized approximation algorithms, and deterministic approximation algorithms. In [19], approximation is achieved by stochastically sampling instantiations of the network variables. Later work in randomized approximation algorithms attempts to increase sampling efficiency [4], and to handle the case where the probability of the evidence is very low [15], which is a serious problem for most sampling algorithms. In what follows, we focus on the second category, *deterministic* approximation algorithms. In *bounded* conditioning [14], one uses the conditioning method, but conditions only on a small, high probability, subset of the (exponential size) set of possible assignments to the cutset variable. Other approximation algorithms attempt to simplify the network by removing arcs between nodes that are *almost* independent, to produce a network that is hopefully tractable topologically. An exact algorithm is then run on the "approximate" network, to produce an approximate answer [22]. Another source of complexity is the large number of states per node in various applications. To alleviate that problem, an approximation based on merging states was suggested [41]. The scheme begins by making all variables unary valued, and successively refining the states of variables, while performing probability updating on the approximate network and thus getting a successively better approximation in each step.

Another category of deterministic approximation algorithms is based on deterministic enumeration of terms or assignments to variables in the network. The idea is to enumerate a set of high-probability complete assignments to all the variables in the network (but frequently partial assignments suffice, see below). The probability of each such assignment can be computed quickly: in O(n), or sometimes even (incrementally) in O(1). The probability of a particular instantiation to a variable v (say $v = v_1$) is approximated by simply dividing the probability mass of all assignments which contain $v = v_1$ by the total mass of enumerated assignments. If only assignments compatible with the evidence are enumerated, this gives the *posterior* probability of $v = v_1$. If the enumerated assignments have a sufficiently large probability mass, then we get a good approximation.

In [12] the ideas of incremental operations for probabilistic reasoning were investigated. Among them was a suggestion for approximating marginal probabilities by enumerating high-probability terms. One interesting point is the skewness result: if a network has a distribution such that every row in the distribution arrays has one entry greater than $\frac{n-1}{n}$, then collecting only $n+1$ assignments, we also have at least $\frac{2}{e}$ of the probability mass. Taking the topology of the network into account, and using term computations, this can presumably be achieved efficiently. However, the skewness assumption as is seems somewhat restrictive. The assumption may hold in some domains, such as circuit fault diagnosis, but not in medical diagnosis, or in the randomly generated networks on which we tested our algorithms. Trying to relax the constraint, say to probability entries greater than $(\frac{n-1}{n})^2$, already forces us to look at $O(n^2)$ assignments to get similar results.

In [28] partial assignments to nodes in the network are created from the root nodes down. The probability of each such assignment is easily computable. Much saving in computational effort is achieved by not bothering about irrelevant nodes, i.e. nodes not above some node that is in the query set, or nodes that are d-separated from the evidence nodes. Later in that paper, an assumption of *extreme probabilities* is made. This is similar to the skewness assumption above. In fact, in the circuit fault diagnosis experiment in [28], the numbers actually used are well within the bounds of the skewness assumption. The algorithm makes use of a conflict scheme in order to narrow the search.

It was already suggested [38, 17] that belief networks frequently have independence structure that is not represented by the topology. Sometimes independence holds given a *particular assignment* to a set of variables V, rather than to all possible assignments to V. In such cases, the topology is no help in determining independence (e.g. d-separation might not hold), the actual distributions might have to be examined. In [38] the idea of independence-based (IB) assignments was introduced. An assignment is a set of (node, value) pairs, which can also be written as a set of node=value instantiations. An assignment is consistent if each node is assigned at most one value. Two assignments are compatible if their union is consistent. Each assignment denotes a (sample space) event, and we thus use the assignment and the event it denotes as synonymous terms whenever this does not lead to ambiguity. An assignment $\mathcal{A}$ subsumes assignment $\mathcal{B}$ if $\mathcal{A} \subseteq \mathcal{B}$.

The *IB condition* holds at a node v w.r.t. an assignment $\mathcal{A}$ if the value assigned to v by $\mathcal{A}$ is independent of all possible assignments to the ancestors of v given $\mathcal{A}_{\text{parents}(v)}$, the assignment made by $\mathcal{A}$ to the immediate predecessors (parents) of v. An assignment is IB if the IB condition holds at every $v \in \text{span}(\mathcal{A})$, where span($\mathcal{A}$) is the set of nodes assigned by $\mathcal{A}$. A ***hypercube*** $\mathcal{H}$ is an assignment to a node v and some of its parents. In this case, we say that $\mathcal{H}$ is *based* on v. $\mathcal{H}$ is an IB hypercube if the IB condition holds at v w.r.t. $\mathcal{H}$.

In [38], IB assignments were the candidates for relevant explanation. Here, we suggest that computing marginal probabilities (whether prior or posterior), can be done by enumerating high-probability IB assignments, rather than complete assignments. Since IB assignments usually have fewer variables assigned, each IB assignment is expected to hold more probability mass than a respective complete (or even a query and evidence supported) assignment. The probability of an IB assignment is also easy to compute [38]:

$$P(\mathcal{A}) = \prod_{v \in \text{span}(\mathcal{A})} P(\mathcal{A}_{\{v\}} | \mathcal{A}_{\text{parents}(v)}) \tag{1}$$

where $\mathcal{A}_S$ is the assignment $\mathcal{A}$ restricted to the set of nodes S. The terms in the product can each be retrieved in O(1) from the conditional distribution array (or other representation) of the node conditional distribution.

One might argue that searching for high-probability assignments for approximating marginal distributions is a bad idea, since coming up with the highest-probability assignment is NP-hard [39]. Thus, we are using an NP-hard algorithm to find an approximate solution to an NP-hard problem, where we might expect that a polynomial time algorithm can be sufficient to compute approximations. However, as noted above, [11] showed that this problem is also NP-hard. Therefore, using this kind of approximation algorithm is a reasonable proposition, provided that some sub-classes of the problem that are bad for existing algorithms can be shown to behave well, either theoretically or by empirical results that show good behaviour on the average for some classes of problems. Since runtimes of our algorithms depend in a complicated manner on the conditional probabilities, it is very hard to get any theoretical bounds on the runtime for interesting classes of networks. In this and related papers, we thus take the experimental route to justify our performance claims.

The rest of the paper is organized as follows: section 2 discusses the details of how to approximate posterior probabilities from a set of high-probability IB assignments. Section 3 reviews the IB MAP search algorithm of [38], and discusses a faster heuristic best-first algorithm for finding the high-probability IB assignments, based on the heuristic presented in [7]. Section 4 reviews the reduction of IB MAP computation to linear systems of equations [38], incorporating a few improvements that reduce the number of equations. Searching for next-best assignments using linear programming is discussed. Section 5 presents experimental timing results for approximation of posterior probabilities on random networks. We conclude with other related work and an evaluation of the IB MAP methods.

2 Computing Marginal Probabilities with IB Assignments

The probability of a certain node instantiation, $v = v_1$, is approximated by the probability mass in the IB assignments containing $v = v_1$ divided by the total mass. If we need to find the probability of v, then v is a *query* node. Nodes where evidence is introduced are called *evidence* nodes. We also assume that the evidence is conjunctive in nature, i.e. it is an assignment of values to the evidence nodes. We need to assume that each enumerated IB assignment $\mathcal{A}$ contains *some* assignment to query node v. Otherwise, it might be impossible to tell which part of the mass of $\mathcal{A}$ supports $v = v_1$. Let us assume for now that this is indeed the case, i.e. we have a set I containing IB assignments, and if $\mathcal{A} \in I$ then $v \in \text{span}(\mathcal{A})$. Thus, to compute the (approximate) probability of $v = v_1$, we compute:

$$P_a(v = v_1) \quad = \quad \frac{P(\{\mathcal{A}|\mathcal{A} \in I \wedge \{v = v_1\} \in \mathcal{A}\})}{P(\{\mathcal{A}|\mathcal{A} \in I\})}$$

Where the probability of a set of assignments is the probability of the event that is the union of all the events standing for all the assignments (not the probability of the union of the assignments). If we are computing the prior probability of $v = v_1$, we can either assume that the denominator is 1 (and not bother about assignments assigning v a value other than v_1), or use $1-P(\{\mathcal{A}|\mathcal{A} \in I\})$ as an error bound. If all IB assignments are disjoint, the probability of the union is easily computable, and is simply the sum of probabilities of the IB assignments.

However, since IB assignments are partial, it is possible for the events denoted by two different IB assignments to overlap. For example, let $\{u, v, w\}$ be nodes, each with a domain $\{1, 2, 3\}$. Then $\mathcal{A} = \{u = 1, v = 2\}$ has an overlap with $\mathcal{B} = \{u = 1, w = 1\}$. The overlap $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ is also an assignment: $\mathcal{C} = \{u = 1, v = 2, w = 3\}$[1]. Thus, to compute the probability of a set, some other method must be used.

Computing the union of the IB assignments in a representation that makes computation of the probabilities easy is non-trivial. It turns out that we can use the inclusion-exclusion principle, due to the following property:

[1]Note that for two assignments $\mathcal{A}, \mathcal{B}$, the *union* of $\mathcal{A}$ and $\mathcal{B}$ denotes the event that is the *intersection* of the events denoted by $\mathcal{A}$ and $\mathcal{B}$.

Theorem 1 *Let $\mathcal{A}, \mathcal{B}$ be compatible IB assignments. Then $\mathcal{A} \cup \mathcal{B}$ is also an IB assignment.*

Proof: Let $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$. Clearly, $\mathcal{C}$ is a consistent assignment. It suffices to show that for each node in $v \in span(\mathcal{C})$, the IB condition holds at v w.r.t. $\mathcal{C}$. Without loss of generality, let $v \in span(\mathcal{A})$. Then v is independent of all its of ancestors given $\mathcal{A}_{parents(v)}$. Now, $\mathcal{A}_{parents(v)} \subseteq \mathcal{C}_{parents(v)}$, and thus, by a variant of weak union ([26] page 84), v is independent of all of its ancestors not in $span(\mathcal{C}) \cap parents(v)$ given $\mathcal{C}_{parents(v)}$.

Despite this theorem, evaluating the probability of a set of IB assignments may require the evaluation of an exponential number of terms. That is due to the equation for implementing the inclusion-exclusion principle:

$$P(\cup_{1 \leq i \leq m} E_i) = \sum_{k=1}^{m} (-1)^{k+1} \sum_{1 \leq a_1 < \ldots < a_k \leq m} \cap_{i=1}^{k} E_{a_i}$$

where E_i is the ith event. Several ways exist to overcome this problem, which we review in short in the discussion section. In the description of the algorithms, this issue is temporarily ignored for simplicity.

How many of the highest probability IB assignments are needed in order to get a good approximation? Obviously, in the worst case the number is exponential in n. However, under the skewness assumption [12] (reviewed in section 1) the number is small. In fact, it follows directly from the skewness theorem [12] that if the highest (or second highest) probability complete assignment is compatible with $\mathcal{A}_{opt}$, the highest probability IB assignment, and $\mathcal{A}_{opt}$ has at least $\log_2 n$ unassigned nodes, then the 2 highest IB assignments contain most ($\geq \frac{2}{e}$) of the probability mass. It is possible to extend the skewness theorem to include $O(n^k)$ terms, in which case the mass will be at least $\frac{T_k(1)}{e}$, where $T_k(x)$ is the polynomial consisting of the first k terms of Taylor expansion of e^x. Thus, under the above conditions, if $\mathcal{A}_{opt}$ has $(k+1)\log_2 n$ unassigned nodes, the highest probability IB assignment will contain at least $\frac{T_k(1)}{e}$ of the probability mass.

Given a set of query and evidence nodes, all non-supported (redundant) nodes can be dropped from the diagram. A node v is supported by a set of nodes V if it is in V or if v is an ancestor of some node in V. A node supported by the evidence nodes is called evidentially supported, and a node supported by a query node is called query supported. We are usually only interested in IB assignments ***properly*** evidentially supported by some set of evidence nodes. An assignment is properly evidentially supported if all the nodes in the assignment have a directed path of ***assigned nodes*** to an evidence node. Likewise, an IB assignment is properly query supported if every node in the assignment obeys the above condition w.r.t. query nodes.

Before we start searching for IB assignments, we drop all evidence nodes that are d-separated from the query nodes, as well as all the nodes that are not either query supported or supported by one of the remaining evidence nodes.

We are now ready do describe the basic anytime best-first search algorithm. The existence of a *generator* is assumed. Each time the generator is called, it returns the next-best (next highest probability) IB assignment consistent with a set of initial assignments. Some variants of the algorithm use more than one generator instance.

- Input: a Bayesian belief network B, evidence $\mathcal{E}$ (a consistent assignment), a query node q.
- Output: successively improved approximations for $P(q = q_i)$, for each value q_i in the domain of node q.

1. Preprocessing
   - Sort the nodes of B such that no node appears after any of its ancestors.
   - Initialize the IB hypercubes for each node $v \in B$.
2. Initializing: remove redundant nodes, and for each q_i in the domain of q do:
   (a) Set up a result set for q_i.
   (b) Add the assignment $\mathcal{E} \cup \{q = q_i\}$ to the initial assignment set for the generator.
3. Repeat until time limit or generator returns null:
   (a) Get next-best IB assignment $\mathcal{A}$ from the generator.
   (b) Add $\mathcal{A}$ to the result set of q_i, where $\{q = q_i\} \in \mathcal{A}$.
   (c) Update the posterior probability approximation.

The simplest generator is a best first search with the current probability heuristic, which is exactly the inner loop of the algorithm in [38], (also described in the following section). In this paper, we also look at two other generators: a best-first search algorithm based on the cost-sharing heuristic, and an integer linear program scheme, modified from [38].

The posterior probability approximation for $q = q_i$ given the evidence is:

$$P_a = \frac{P(\text{result set for } q_i)}{\sum_i P(\text{result set for } q_i)}$$

As before, for null evidence, $1 - \sum_i P(\text{result set for } q_i)$ is the unassigned probability mass, and can be used to bound the error, as in [28]. In order to bound the error for non-null evidence, we evaluate the probability that the evidence is false by using the same scheme. That is, add to the initialization set the assignment FALSE to the evidence node[2], and create a result set for it.

Note that the preprocessing need only be done once for each network, and the results can be used for different query and evidence sets. Alternately, it is possible to do most of the preprocessing incrementally by moving it into the loop, and initialize the hypercubes for a node only when we first try to expand it (i.e. inside the generator). This way, the algorithm can sometimes start providing answers before initializing all the hypercubes. In fact, it is not even necessary that the belief network be explicitly represented in entirety. Applications which construct belief networks incrementally (such as WIMP [5]) might benefit from not having to generate parts of the network unless needed for abductive conclusions.

A variant of the algorithm uses several generators, one for each assignment in the above described initialization sets (thus each generator now gets an initialization set of size 1). Thus, there is one generator for the negation of the evidence, and one generator for each state of the query node. In each approximation step, get one next-best IB assignment from each generator, and proceed to update the marginal probability estimate. We believe that this version of the algorithm achieves a better balance for the case where the posterior probability of some state of the query node is low, and thus a better relative approximation. This issue is orthogonal to the actual algorithm used to find the IB assignments, and is ignored henceforth.

To generalize this algorithm to m query nodes, it is sufficient to initialize a result set for every state in the domain of each node (*not* the cross product, which is what we would so if we wanted to find the posterior *joint* probability of the query nodes), and to add to the initialization set an assignment for each such state. When an IB assignment is found, it is added into m result sets, one for each query node. To evaluate the probability approximation, divide by the probability of the set of assignments collected for only one of the nodes (any one will do).

Experimental results from [38] suggest that at least the highest probability IB assignment (the IB-MAP) can be found in reasonable time for medium-size networks (up to 100 nodes), but that problems start occurring for many instances of larger networks. Experiments for finding several of the next-best IB assignments are reviewed in section 5, and timing results tend to indicate that next-best assignments are found rather quickly after the first one. However, we would still like to see a faster algorithm. The method of using IB assignments to approximate posterior probabilities can be divorced from the search method (the generator). Any generator providing the IB assignments in the correct

[2]We can assume without loss of generality that the evidence consists of a single binary valued node. Otherwise, create a new, deterministic, binary valued node with all the evidence nodes as parents, with the requisite function (usually the new node is TRUE if all parents are instantiated as in the evidence, FALSE otherwise, i.e. the new node is a generalized AND node).

order will do. In the next section, we discuss how the linear program techniques used in [35, 38] can be used to deliver IB assignments in decreasing order of probability, for posterior probability approximation.

3 Heuristic Search IB-Assignment Generators

In this section, we present best-first heuristic search generators for the marginal probability approximation algorithms. We begin with the simple, current probability heuristic algorithm [38], and then discuss the better cost-sharing heuristic. The best-first algorithm keeps a sorted agenda of states, where a state is an assignment, a node last expanded, and a probability estimate:

- Input: a Bayesian belief network B, a set of consistent assignments $\mathbf{E}$
- Output: The next best IB assignment, consistent with some $\mathcal{E} \in \mathbf{E}$

1. Initializing: for each $\mathcal{E}$ in $\mathbf{E}$, push into the agenda the assignment $\mathcal{E}$ with a probability estimate of 1.
2. Repeat until empty agenda:
   (a) Pop assignment with highest estimate $\mathcal{A}$ from the agenda, and remove duplicate assignments (they will all be at the top of the agenda).
   (b) If $\mathcal{A}$ is IB, return it.
   (c) Otherwise, expand $\mathcal{A}$ at v, the next node, into a set of assignments $\mathcal{S}$, and for each assignment $\mathcal{A}^j \in \mathcal{S}$ do:
      i. Estimate the probability of $\mathcal{A}^j$.
      ii. Push $\mathcal{A}^j$ with its probability estimate and last-expanded node v into the agenda.

When the generator is resumed (i.e. called after it returns the first time), the resumption point is at step 2. Expanding a state and the probability estimate is exactly as in [38]: $\mathcal{A}^j = \mathcal{A} \cup \mathcal{H}^j$, where $\mathcal{H}^j$ is the jth IB hypercube based on v that is maximal w.r.t. subsumption and consistent with $\mathcal{A}$. The probability estimate is the product of hypercube probabilities for all nodes were the IB condition holds.

We now consider improving the performance of our search algorithm by using a different heuristic than current probability (which is essentially the same as "cost-so-far" [40]). The idea is that cost-so-far gives little information early on in the search, while including costs that will be incurred later on (higher up in the DAG) give us a better estimate. However, one cannot just add the costs to be incurred in the future, because in multiply connected networks one node cost (negative logarithm of probability) would be counted multiple times, and we would no longer have an admissible heuristic. The idea of dividing the cost to be incurred by the number of children, the "cost-sharing" heuristic, was pursued in [7] for proof graphs (AND/OR DAGs). The cost sharing heuristic showed a marked improvement in performance over the cost-so-far heuristic when applied to graphs generated by WIMP [6]. Since the above generator is a best-first search algorithm that uses the cost-so-far heuristic, plugging in the cost-sharing heuristic ought to give us a great improvement in performance, assuming that it can be applied to IB assignment search.

The cost-sharing heuristic is admissible only if the expansion operator is over edges (rather than nodes), and obeys the *minimal cut* property. A cut of an AND DAG (a directed acyclic graph containing only AND nodes) is a set of edges E such that every path from any root node to a leaf node contains an edge from E. A cut is minimal if it is setwise minimal, i.e. if no edge can be removed from E such that there is still a cut. (what we call a "minimal cut" here is the same as a "cut" in [7].) For an AND/OR DAG D, E is a cut if D contains a complete AND DAG (intuitively: completely specified proof) for which E is a cut. Likewise for a minimal cut of an AND/OR DAG.

In order to use the heuristic, we must first convert our problem into a weighted AND/OR DAG (WAODAG), and then provide an edge-based operator that preserves the minimal cut property. We must also show that probabilities are equivalent to costs in the WAODAG. If we do all of the above, we are assured by the results of [7], that the heuristic is admissible for our search, and that this algorithm variant indeed comes up with the *highest-probability* IB assignment.

To convert our problem into the WAODAG formulation, we perform a construction similar to [8] or [40]. The algorithm is given a belief network $B = (V, A, P)$, and evidence $\mathcal{E}$.[3] We construct a WAODAG $W(B, \mathcal{E}) = (G, c, f, S)$ that is equivalent to the original belief network and evidence. In a WAODAG, G is the underlying DAG, f is the *label* of a node, which is either AND or OR, c is the cost function, and S is the sink node. The construction is as follows, except for the costs, which are discussed later on:

- For each possible node-state $(v, d) \in \mathcal{E} \cup \{(v, d) \mid v \in V - \text{span}(\mathcal{E}) d \in D_v\}$ construct a OR node N^{v^d}. Note that for each evidence node, only one state is possible.

- Construct an AND node S, with parents N^{v^d} for all $(v, d) \in \mathcal{E}$, i.e. all the evidence node-states.

- For each maximal IB hypercube based on any node v that assigns value d to v, construct an AND node $H_i^{v^d}$, where i is the index of the hypercube among all the hypercubes that are based on v and assign a value d to v (the actual order is immaterial). We call $H_i^{v^d}$ the node image of the

[3]Note that query nodes essentially become evidence nodes, in the context of searching for the best IB assignment. Additionally, we can assume without loss of generality that all nodes are either evidence or query nodes, or ancestors of some such node (otherwise they can just be dropped from the diagram).

hypercube, and use it as a synonym for the hypercube itself whenever unambiguous.

- For each maximal IB hypercube as above, construct a node $SC_i^{v^d}$, the hypercube's "self cost" node.

- Construct a directed edge from each hypercube $H_i^{v^d}$ to N^{v^d}.

- Each hypercube assigns some state to some of a node's parents. Let $A_i^{v^d}$ be the set of node-state pairs in hypercube $H_i^{v^d}$. Construct an edge from each node $N^{w^{d'}} \in A_i^{v^d}$ to $H_i^{v^d}$.

- From each self-cost node $SC_i^{v^d}$ construct a directed edge $ESC_i^{v^d}$ (self-cost edge) to the respective $H_i^{v^d}$. Node $SC_i^{v^d}$ can be viewed as an AND node with no parents.

Let AD be a complete AND DAG in W, and let $\mathcal{H}$ be the set of hypercubes $H_i^{v^d}$ in AD. Define $a(AD)$ to be the assignment consisting of the union of all the hypercubes $H_i^{v^d} \in \mathcal{H}$. Let C be a (minimal) cut of W consisting only of the self-cost edges of a set of hypercubes $\mathcal{H}$. As before, define $a(C)$ as the assignment consisting of the union of all the hypercubes in $\mathcal{H}$. Likewise, if C is a set of edges (not necessarily a cut or a minimal cut), define $a(C)$ to be the union of all the hypercubes for which their self-cost edge is in C.

We now complete the definition of the WAODAG by defining the cost function. The cost of a self-cost node $c(SC_i^{v^d}) = -\log P(H_i^{v^d})$. The rest of the costs are defined as in [7], as follows:

- The cost of a self-cost edge is the cost of its source node, i.e. $c(ESC_i^{v^d}) = c(SC_i^{v^d}) = -\log P(H_i^{v^d})$.

- The cost of a hypercube $c(H_i^{v^d})$ is the sum of the costs of the incoming edges.

- The cost of each edge with a hypercube node as a source is the cost of its source hypercube.

- The cost of any other edge is the cost of its source node N^{v^d} divided by the number of children that v has in B.

- The cost of a node-state node N^{v^d} is the minimum of the costs of all of its incoming edges.

Since W is a DAG, and the above defines costs only in terms of the belief network or edges and nodes above, this definition is well grounded (i.e. is not circular).

The generator keeps an agenda of states, where a state is a set of edges, a minimal cut C. For convenience and efficiency, we also keep the hypercubes, last expanded node, current heuristic value, etc. but these can all be computed from the cut C. The heuristic value of a state is the sum of its edge costs. Our expansion operator σ (a function from a set of edges to a set of sets of edges, i.e. essentially from a state to a set of next states) is the same as in [7], except that when an OR node is expanded to include an edge from a hypercube $H_i^{v^d}$ to N^{v^d}, we expand the hypercube node (which is an AND node) as well in the same expansion step. This does not affect either the heuristic value or the reachability of final states. Hence, to prove that this heuristic is admissible, the results of [7] can be directly applied. We begin by presenting the algorithm for searching for IB MAP. Its extension to compute next best IB assignments and posterior probabilities is exactly the same as for the cost-so-far heuristic search. To use this algorithm as a generator, we assume that there is only one assignment in the initialization set, and call it the evidence.

- Input: a Bayesian belief network B, and a consistent assignment $\mathcal{E}$, the evidence.
- Output: Next best IB assignment.

1. Initializing
   (a) Remove redundant nodes.
   (b) Create the WAODAG from the top down, while computing node and edge costs.
   (c) Create a dummy outgoing edge e from S, with cost equal to $c(S)$ (the dummy "edge" does not even need to have a sink).
   (d) Push the singleton set $\{e\}$ onto the agenda, with heuristic cost equal to $c(e)$.
2. Repeat until empty agenda:
   (a) Pop state with lowest cost estimate C from the agenda, and remove duplicate states (they will all be at the top of the agenda).
   (b) If the assignment is IB (all edges are self cost edges), return it.
   (c) Otherwise, find the earliest node N^{v^d} which is a source of an edge e in C, and compute $\sigma(C)$. That is, expand C at e (we also call this "expanding at node v") into a set of states $\mathcal{C}$. For each state $C' \in \mathcal{C}$ do the following:
      i. Find if C' corresponds to a consistent assignment, and if not, discard it.
      ii. Compute the heuristic value for C' as the sum of edge costs.

iii. Push C' into the agenda.

Expanding a state C at e with source N^{v^d} (computation of σ) works as follows: Let E be the set of edges with source N^{v^d} in C. The parents of N^{v^d} are $H_i^{v^d}$ with $1 \leq i \leq$ indegree($N_i^{v^d}$). Then the new states C_i, $1 \leq i \leq$ indegree(N^{v^d}) are $C_i = C - E +$ incoming($H_i^{v^d}$). Note that by construction, each application of the expansion adds one self-cost edge to C, thereby adding a hypercube to the assignment defined by the cut. Each of the new states amounts to a different choice of hypercube at v, just like the cost-so far algorithm. In [33], we show that the algorithm is correct.

This algorithm diverges from that of [7] in that when an edge from an OR node is expanded, the AND node which is its parent is also expanded immediately, in the same expansion step. This does not affect the correctness of the algorithm, since for all consistent self-cost-only cuts C either it is reachable, or $a(C)$ is subsumed by some IB assignment $a(C')$, where C' is reachable:

The algorithm also diverges from [7] in that the cost of an edge is the cost of its source node N^{v^d} divided by the out degree of v in B, rather than the out degree of N^{v^d} in W. That is permissible because of all the nodes $H_i^{u^{d'}}$ (where u is a child of v in B, and for any value d' and hypercube index i) only one appears in any AND DAG, and thus the cost is only shared among at most outdegree(v) edges. This argument is similar to the one presented in the conclusion of [7].

Note that it is possible for an edge e with source v to exist in C, where actually the IB condition holds at v w.r.t. $a(C)$. In this case, we still need to expand e, but it does not matter which next state is selected, they all result in the same assignment (counting only consistent assignments). In the actual implementation, to prevent the creation of too many duplicate states, the first one of these that is consistent is pushed into the agenda, and all the others are discarded. While the selection of edges may affect the search in that the cost estimation may be different, it cannot affect the final result in this case.

Finally, how is this generator to be used in the marginal probability approximation algorithm? In the variant where each generator gets a singleton set as an initialization set, the generator need no modification. In the variant where one generator is used for all states, only minor modifications are needed, as follows. First, we need to create S_i, a copy of the node S, for each assignment $\mathcal{E}_i$ in the initialization set, and create the WAODAG accordingly. One dummy edge e_i is created for each node S_i, and one agenda item is created initially for each $\mathcal{E}_i$. No further modifications are necessary. Something is lost by the fact that to find assignments consistent with just the negation of the original evidence node (used for bounding the error in the approximation algorithm), we do not need any predecessors of the query nodes. The cost-sharing heuristic suffers somewhat as a result (even though it is still admissible), as it is less optimistic. To avoid this problem, one can always just use the multi-generator version of the algorithm.

4 REDUCTION TO ILP

In [32], [35], [34], and [31], a method of converting the complete MAP problem to an integer linear program (ILP) was shown. In [38] a similar method that converts the problem of finding the IB MAP to a linear inequality system was shown. We begin by reviewing the reduction, which is modified somewhat from [38] in order to decrease the number of equations, and discuss the further changes necessary to make the system find the next-best IB assignments.

The linear system of inequalities has a variable for each maximal IB hypercube. The inequality generation is reviewed below. A belief network is denoted by $B = (G, \mathcal{D})$, where G is the underlying graph and $\mathcal{D}$ the distribution. We usually omit reference to $\mathcal{D}$ and assume that all discussion is with respect to the same arbitrary distribution. For each node v and value in D_v (the domain of v), there is a set of k_{v^d} maximal IB hypercubes based on v (where $d \in D_v$). We denote that set by $\mathcal{H}^{v^d}$, and assume some indexing on the set. Member j of $\mathcal{H}^{v^d}$ is denoted $\mathcal{H}^{v^d}_j$, with $k_{v^d} \geq j \geq 1$.

A system of inequalities L is a triple (V, I, c), where V is a set of variables, I is a set of inequalities, and c is an assignment cost function.

Definition 1 *From the belief network B and the evidence $\mathcal{E}$, we construct a system of inequalities $L = L_{IB}(B, \mathcal{E})$ as follows:*

1. *V is a set of variables $h^{v^d}_i$, indexed by the set of all evidentially supported maximal hypercubes $H_\mathcal{E}$ (the set of hypercubes H such that if H is based on w, then w is evidentially supported). Thus, $V = \{h^{v^d}_i | H^{v^d}_i \in H_\mathcal{E}\}$.*[4]*,*

2. *$c(h^{v^d}_i, 1) = -log(P(H^{v^d}_i))$, and $c(h^{v^d}_i, 0) = 0$.*

3. *I is the following collection of inequalities:*

   (a) *For each triple of nodes (v, x, y) s.t. $x \neq y$ and $v \in parents(x) \cap parents(y)$, and for each $d \in D_v$:*

$$\sum_{(v,d) \in H^{x^e}_i, e \in D_x} h^{x^e}_i + \sum_{(v,d') \in H^{y^f}_j, f \in D_y, d \neq d'} h^{y^f}_j \leq 1 \tag{2}$$

   (b) *For each evidentially supported node v that is not a query node and is not in $span(\mathcal{E})$:*

$$\sum_{d \in D_v} \sum_{i=1}^{k_{v^d}} h^{v^d}_i \leq 1 \tag{3}$$

[4]The superscript v^d states that node v is assigned value d by the hypercube (which is based on v), and the subscript i states that this is the ith hypercube among the hypercubes based on v that assign the value d to v.

(c) *For each pair of nodes w, v such that $v \in parents(w)$, and for each value $d \in D_v$:*

$$\sum_{i=1}^{k_{vd}} h_i^{v^d} - \sum_{d' \in D_w \wedge (v,d) \in H_j^{w^{d'}}} h_i^{w^{d'}} \geq 0 \tag{4}$$

(d) *For each $(v, d) \in \mathcal{E}$:*

$$\sum_{i=1}^{k_{vd}} h_i^{v^d} = 1 \tag{5}$$

(e) *For each query node q:*

$$\sum_{d \in D_q} \sum_{i=1}^{k_{qd}} h_i^{q^d} = 1 \tag{6}$$

The intuition behind these inequalities is as follows: inequalities of type **a** enforce consistency of the solution. Type **b** inequalities enforce selection of at most a single hypercube based on each node. Type c inequalities enforce the IB constraint, i.e. at least one hypercube based on v must be selected if v is assigned. Type **d** inequalities introduce the evidence, and type e introduces the query nodes. Modifications from [38] include imploding several type **a** equations into one, reducing the number of such equations by roughly a factor quadratic in the number of hypercubes per node. Other modifications are making type **b** and **d** into equalities to make a simpler system, and adding the equations for the previously unsupported query nodes.

Following [32], we define an assignment s for the variables of L as a function from V to $\mathcal{R}$. Furthermore:

1. If the range of s is in $\{0, 1\}$ then s is a 0-1 assignment.
2. If s satisfies all the inequalities of types **a-d** then s is a solution for L.
3. If solution s for L is a 0-1 assignment, then it is a 0-1 solution for L.

The objective function to optimize is:

$$\Theta_{L_{IB}}(s) = -\sum_{h_i^{v^d}} s(h_i^{v^d}) log(P(H_i^{v^d})) \tag{7}$$

In [38] it was shown that a optimal 0-1 solution to the system of inequalities induces an IB MAP on the original belief network. The minor modifications introduced here, while having a favorable effect on the complexity, encode the same constraint and this do not affect the problem equivalence results of [38].

If the optimal solution of the system happens to be 0-1, we have found the IB MAP. Otherwise, we need to branch: select a variable h which is assigned a non 0,1 value, and create two sets of inequalities (subproblems), one with $h = 1$ and the other with $h = 0$. Each of these now needs to be solved for an optimal 0-1 solution, as in [35]. This branch and bound algorithm may have to solve an exponential number of systems, but in practice that is not the case. Additionally, the subproblems are always smaller in number of equations or number of variables.

To create a subproblem, h is *clamped* to either 0 or 1. The equations can now be further simplified: a variable clamped to 0 can be removed from the system. For a variable clamped to 1, the following reductions take place: Find the type **b** inequality, the type **d** equation (if any), and all the type **a** inequalities, in which h appears. In each such inequality clamp all the other variables to 0 (removing them from the system), and delete the inequality. After doing the above, check to see if any inequality contains only one variable, and if so clamp it accordingly. For example, if a type **d** equation has only one variable, clamp it to 1. Repeat these operations until no more reductions can be made.

Once the optimal 0-1 solution is found, we need to add an equation prohibiting that solution, and then to find an optimal solution to the resulting set of equations.

Let S be the set of nodes in the IB assignment $\mathcal{A}$ induced by the optimal 0-1 solution. To update the system, add the following equation:

$$\sum_{v \in S} \sum_{\{H_i^{v^d} | (v,d) \in \mathcal{A}\}} h_i^{v^d} < |S|$$

This equation prevents any solution which induces an assignment $\mathcal{B}$ s.t. the variables in S are assigned the same values as in $\mathcal{A}$. Thus, it is not just a recurrence of $\mathcal{A}$ that is prohibited, but of any assignment $\mathcal{B}$ subsumed by $\mathcal{A}$, in which case we would *also* like to ignore $\mathcal{B}$.

5 Preliminary Experimental Results

As shown in experiments in [38], finding highest probability IB assignments is feasible for up to medium-size diagrams, even with the current probability heuristic. However, the method begins to deteriorate rapidly starting at 100 nodes. Hence, we turn to the cost sharing and linear programming approaches.

Preliminary results show that our constraints approach can solve for the IB MAP in networks of up to 2000 node. Figure 1 compares the timing results of the linear programming approach on 50 networks each consisting of 2000 nodes, with the current cost and shared cost methods. For these problem instances, cost sharing usually did much better than ILP, which in turn did much better than current cost. However, we expect that on larger diagram sizes, ILP will

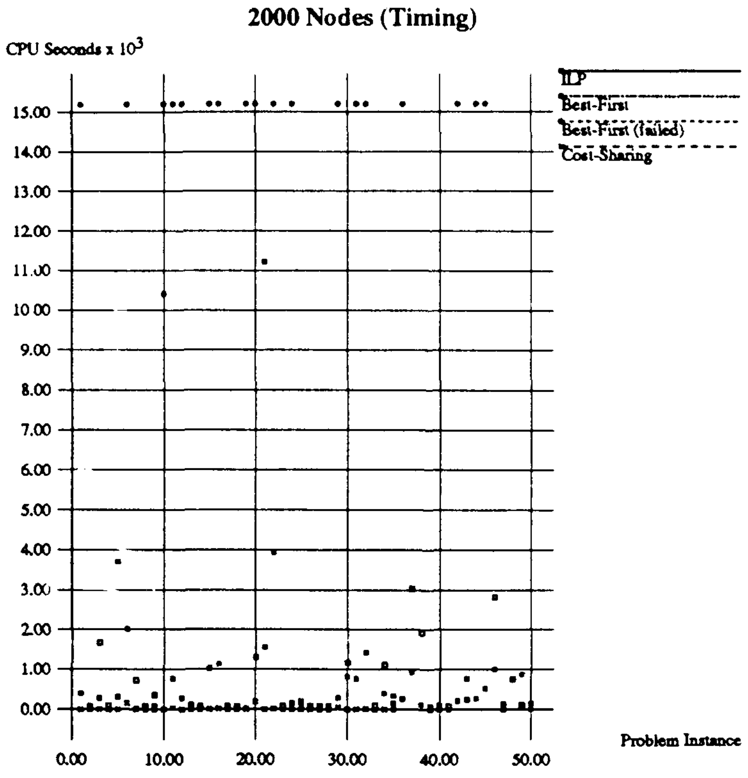

Figure 1: 2000 node networks.

do better than cost sharing, which we intend to confirm in the near future.[5] For the most part, we found our ILP solutions relatively quickly. We would like to note though, that our package for solving integer linear programs was crudely constructed by the authors without the additional optimizations such as sparse systems, etc. Furthermore, much of our computational process is naturally parallelizable and should benefit immensely from techniques such as parallel simplex [18] and parallel ILP [2, 3].

[5]There was one problem instance (not shown) for which cost-sharing took so long (and so much memory) that it crashed the lisp interpreter (which also happened for *several* of the "failed" cases for current cost).

6 Discussion

Having presented several IB assignment generators, we would now like to tie the loose ends together by referring to the problem of overlapping IB assignments. This section also addresses the applicability of the method, i.e. in what type of networks does the use of instantiation-based independence buy us anything. We conclude this section by a comparison to related work.

6.1 Treatment of Overlapping Assignments

There are several ways to handle overlapping assignments, ranging from avoiding overlaps, to approximating the inclusion-exclusion formula. Poole suggests avoiding this problem altogether by not generating any overlaps in the explanation extension stage. In [27], marginal probabilities are computed by adding up the probability mass in disjoint ***explanations***, which are a special case of IB assignments. Disjointness is achieved by, whenever generating several extended explanations from a as single partial explanations, making sure that the extended explanations are disjoint. To do that, whenever a set of candidate explanations A is considered for a proposition b (where the database has rules of the form $b \leftarrow a_i$, for each $a_i \in A$), the extended explanations will consist of the set:

$$\{\{b, a_1\}, \{b, a_2, \neg a_1\}, ..., \{b, a_{|A|}, \neg a_1, ..., \neg a_{|A|-1}\}\}$$

in which all the explanations are clearly pairwise (as well as globally) disjoint. Unfortunately, when translated into Bayes net format, such rule sets are OR nodes, and it is not clear how to handle other types of nodes. Poole does explain how a Bayes net might be represented in this scheme. It turns out, however, that negating a variable in the explanation maps into a non-trivial constraint in the network. In addition, it may not be desirable to do this anyway. What if the explanation $\{b, a_{|A|}\}$ turns out to be by far the most likely overall? In Poole's scheme, we have eliminated much of its probability mass early on, and there is no reasonable way to reorder the explanations to get, say:

$$\{\{b, a_{|A|}\}, \{b, a, \neg a_{|A|}\}, ...\}$$

The way Poole's algorithm works, the ordering is imposed based on some heuristic (e.g. the causal strength of the rules), but there is no guarantee that what appears best at mid-run will indeed be a global optimum. For these reasons, we do not employ the above method of avoiding overlaps. Our solutions are based on ***defering*** the decision as to which explanation (IB assignment) is best, and then (if necessary) preventing ***successive*** IB assignments from overlapping previous ones. At the stage of the algorithm where we get the IB assignments, we know their probability ordering exactly. It is possible to defer computation of high order higher-order terms in the inclusion-exclusion formula. The probability of these terms diminishes, and we could ignore them in the computation.

That is because low-probability assignments are going to be ignored in the approximation algorithm anyway. Theoretically, this is a bad idea. As shown in [25], we need a very large number of terms (about $2^{\sqrt{n}}$ in the worst case) to get a good approximation of the inclusion-exclusion formula, in the general case. Still, this might be feasible in a practical algorithm.

Nevertheless, we use inclusion-exclusion only for a *small* set of overlapping assignments, and prevent the occurrence of sets of overlapping assignments with cardinality strictly greater than some small integer constant k. The exact value of of k depends on which algorithm variant we use. In the best-first heuristic search algorithms, it is hard to prevent an IB assignment from overlapping all other assignments, and we thus use $k = 3$. If an IB assignment $\mathcal{A}$ comes up that is not subsumed by some previously generated IB assignment (in which case it is thrown out), we do the following test. If $\mathcal{A}$ overlaps more than k IB assignments, we split it into several assignments (which are not necessarily IB anymore), and toss the new assignments back into the agenda. The details of this method are examined elsewhere [33]. In the ILP version of the algorithm, it is much easier to prevent an overlap, and thus we can use $k = 0$, which means that no overlapping IB assignments are generated. See [33] for details.

6.2 Compactness of Hypercube Representation

Central to all of the algorithms for approximating marginal probabilities by enumerating IB assignments is the number of maximal IB hypercubes representing a conditional distribution. The number of assignments generated in each step of the loop for the search-based algorithm is some fraction of the number of hypercubes per node. In the ILP scheme, the number of equations as well as the number of terms per equation also depends on the number of hypercubes. The issue of how many hypercubes are needed to represent a conditional distribution in a network is therefore paramount.

In our experiments, random network distributions were generated by splitting a hypercube into subcubes with some probability p. One may ask if this represents a typical case of Bayesian networks in applications. We noted in [38] that dirty OR nodes (as well as pure OR nodes, AND nodes, etc.) are compactly representable as maximal IB hypercubes ($2k$ hypercubes for a k parent dirty OR node). However, a much more commonly encountered type of node is the *noisy* OR. It turns out that if a noisy OR is represented as a single node (with a single distribution array), its maximal hypercube representation is of size 2^k, which is certainly *not* compact.

If we use δ-IB hypercubes [38], and the noisy OR has high causal strength links, again $2k$ maximal δ-IB hypercubes will suffice. However, it is unclear how δ-IB assignments can be used for approximating marginal probabilities without severely impairing the precision of the approximation algorithm (the probability of a δ-IB assignment can only be *approximated* in linear time, but not computed exactly). It turns out, however, that representing a noisy OR in its causal independence form [1, 26], with extra nodes added between the causes and the noisy OR node itself, things are much improved. In this representation, the noisy OR is translated into a pure OR, and can be represented with $2k$ hypercubes. The additional nodes require another $4k$ hypercubes. And, in fact, if zero probability hypercubes are dropped (they will never participate in any useful IB assignment), a total of $3k$ hypercubes suffice to describe a noisy OR.

Since the commonly used noisy OR is compactly representable, it is interesting whether other cases of causal independence can be represented compactly, such as noisy MAX [29], as well as more general types of causal independence. For deterministic binary valued nodes, this question reduces to the question: is there a compact DNF representation for the function? Likewise, in the causal independence representation, the deterministic part is separated out as a deterministic function f and k noisy "channels". If f is compactly representable in DNF, then the answer is affirmative. In the more general case of multi-valued nodes, the issue is a bit more complicated, and is beyond the scope of this paper. Suffice it to say that the Generalized IB hypercubes presented in [37] can be used to better advantage with multi-valued nodes, since they allow the aggregation of values in a node into a single "macro value".

6.3 Related and Future Work

The work on *term computation* [12] and related papers are extremely relevant to this paper. The skewness assumption made there, or a weaker version of it, also make our method applicable. In a sense, these methods complement each other, and it should be interesting to see whether IB assignments (or at least maximal IB hypercubes) can be incorporated into a term computation scheme.

This paper enumerates high probability IB assignments using a backward search from the evidence. [28] also enumerates high probability assignments, but using a top down (forward) search. Backward constraints are introduced through conflicts. It is clear that the method is efficient for the example domain (circuit fault analysis), but it is less than certain whether other domains would obey the extreme probability assumption that makes this work. If that assumption does not hold, it may turn out that backward search is still better. On the other hand, it should be possible to take advantage of IB hypercubes even in the forward search approach [33]. Note that among the algorithms presented here, the current probability heuristic ignores forward constraints, while the shared-cost heuristic does employ some form of forward reasoning by incorporating the costs from the top-down initialization. The ILP method uses global constraints that also include the top-down constraints, but what role the top-down constraints play in the search is unclear.

Several stochastic approximation algorithms find the MAP. For example, in [16] simulated annealing is used. It is not clear, however, how one might use it either to enumerate a number of high-probability assignments or make it search for the IB MAP. A genetic algorithm for finding the MAP [30] makes a more interesting case. The authors in [30] note that the probability mass of the population rises during the search and converges on some value. They do not say whether assignments in the population include duplicates, however, and make no mention of the possibility of approximating marginal probabilities with that population. It seems likely that if the search can be modified to search among IB assignments, then the fact that a whole *population* is used, rather than a single candidate, may provide a ready source of near-optimal IB assignments. Of course, we are not guaranteed to get IB assignments in decreasing order of probability, so slightly different methods would have to be used to approximate the marginal probabilities.

Finally, it should be possible to modify the algorithms presented in this paper to work on GIB assignments and δ-IB assignments, where an even greater probability mass is packed into an assignment [38, 37]. Some theoretical issues will have to be resolved before we can do that, however.

7 SUMMARY

Computing marginal (prior or posterior) probabilities in belief networks is hard. Approximation schemes are thus of interest. Several deterministic approximation schemes enumerate terms, or assignments to sets of variables, of high probability, such that a relatively small number of them contain most of the probability mass. This allows for an anytime approximation algorithm, whereby the approximation improves as a larger number of terms is collected. IB assignments are partial assignments that take advantage of local independencies not represented by the topology of the network, to reduce the number of assigned variables, and hence the probability mass in each assignment.

What remains to be done is to come up with these IB assignments in a decreasing order of probability. This is also a hard problem in general, unfortunately. The factors contributing to complexity, however, are not maximum clique size or loop cutset, but rather the number of hypercubes. Under probability skewness assumptions, the search for high probability IB assignments is typically more efficient, and the resulting approximation (collecting a small number of assignments) is better.

Three algorithms for approximating marginal algorithms are presented: a modification of a node-based best-first search algorithm for finding the IB MAP, an edge-based best-first search algorithm with a cost-sharing heuristic, and an algorithm based on linear systems of inequalities. We have also experimented on highly connected diagrams were the conditional probabilities are represented as sets of hypercubes (distribution arrays are precluded, since they are exponential in size), and got favorable results in cases where the join-tree algorithm cannot handle in practice.

Preliminary results show that using the cost-sharing heuristic improves performance of the best-first search algorithm by more than one order of magnitude, and that the algorithm based on linear systems of inequalities is still faster. Naturally, more conclusive experiments will have to be performed.

References

[1] A New Look at Causal Independence. David heckerman and john s. breese. In *Uncertainty in AI, Proceedings of the Tenth Conference*, pages 286–292, July 1994.

[2] Paul D. Bailor and Walter D. Seward. A distributed computer algorithm for solving integer linear programming problems. In *Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers and Applications*, pages 1083–1088, 1989.

[3] Rochelle L. Boehning, Ralph M. Butler, and Billy E. Gillett. A parallel integer linear programming algorithm. *European Journal of Operational Research*, 34:393–398, 1988.

[4] Remco R. Bouckaert. A stratified simulation scheme for inference in Bayesian belief networks. In *Uncertainty in AI, Proceedings of the Tenth Conference*, pages 110–117, July 1994.

[5] Eugene Charniak and Robert Goldman. A logic for semantic interpretation. In *Proceedings of the 26th Conference of the ACL*, 1988.

[6] Eugene Charniak and Robert P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 1994.

[7] Eugene Charniak and Saadia Husain. A new admissible heuristic for minimal-cost proofs. In *Proceedings of AAAI Conference*, pages 446–451, 1991.

[8] Eugene Charniak and Solomon E. Shimony. Cost-based abduction and MAP explanation. *Artificial Intelligence Journal*, 66:345–374, 1994.

[9] Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42 (2-3):393–405, 1990.

[10] Gregory Floyd Cooper. *NESTOR: A Computer-Based Medical Diagnosis Aid that Integrates Causal and Probabilistic Knowledge.* PhD thesis, Stanford University, 1984.

[11] Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60 (1):141–153, 1993.

[12] Bruce D'Ambrosio. Incremental probabilistic inference. In *Uncertainty in AI, Proceedings of the 9th Conference*, July 1993.

[13] F. J. Diez. Local conditioning in Bayesian networks. Technical Report R-181, Cognitive Systems Lab., Dept. of Computer Science, UCLA, July 1992.

[14] Gregory F. Cooper Eric J. Horvitz, H. Jacques Suermondt. Bounded conditioning: Flexible inference for decisions under scarce resources. In *5th Workshop on Uncertainty in AI*, August 1989.

[15] Robert Fung and Brendan Del Favero. Backward simulation in bayesian networks. In *Uncertainty in AI, Proceedings of the Tenth Conference*, pages 227–234, July 1994.

[16] Stuart Geeman and Donald Geeman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[17] Dan Geiger and David Heckerman. Advances in probabilistic reasoning. In *Proceedings of the 7th Conference on Uncertainty in AI*, 1991.

[18] B. E. Gillett. *Introduction to Operations Research: A Computer-Oriented Algorithmic Approach.* McGraw Hill, 1976.

[19] J. Y. Halpern and R. Fagin. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings Uncertainty in Artificial Intelligence 4*, pages 149–163, 1988.

[20] F. V. Jensen, K. G. Olsen, and S. K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20:637–660, 1990.

[21] Jin H. Kim and Judea Pearl. A computation model for causal and diagnostic reasoning in inference systems. In *Proceedings of the 6th International Joint Conference on AI*, 1983.

[22] Uffe Kjaerulff. Reduction of computational complexity in Bayesian networks through removal of weak dependencies. In *Uncertainty in AI, Proceedings of the Tenth Conference*, pages 374–382, July 1994.

[23] S.L. Lauritzen and David J. Speigelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistical Society*, 50:157–224, 1988.

[24] Z. Li and Bruce D'Ambrosio. An efficient approach to probabilistic inference in belief nets. In *Proceedings of the Annual Canadian AI Conference*, May 1992.

[25] Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10:349–365, 1990.

[26] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[27] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[28] David Poole. The use of conflicts in searching Bayesian networks. In *Uncertainty in AI, Proceedings of the 9th Conference*, July 1993.

[29] Malcolm Pradhan, Gregory Provan, Blackford Middleton, and Max Henrion. Knowledge engineering for large belief networks. In *Uncertainty in AI, Proceedings of the Tenth Conference*, pages 484–490, July 1994.

[30] Carlos Rojas-Guzman and Mark A. Kramer. GALGO: A Genetic ALGOrithm decision support tool for complex uncertain systems modeled with Bayesian belief networks. In *Uncertainty in AI: Proceedings of the 9th Conference*. Morgan Kaufmann, 1993.

[31] Eugene Santos, Jr. A fast hill-climbing approach without an energy function for finding mpe. In *Proceedings of the 5th IEEE International Conference on Tools with Artificial Intelligence*, 1993.

[32] Eugene Santos, Jr. A linear constraint satisfaction approach to cost-based abduction. *Artificial Intelligence*, 65(1):1–28, 1994.

[33] Eugene Santos, Jr. and Solomon E. Shimony. An empirical study on deterministic approximation of marginal probabilities in Bayes nets. *Submitted to JETAI*, 1994.

[34] Eugene Santos Jr. Cost-based abduction, linear constraint satisfaction, and alternative explanations. In *Proceedings of the AAAI Workshop on Abduction*, 1991.

[35] Eugene Santos Jr. On the generation of alternative explanations with implications for belief revision. In *Proceedings of the 7th Conference on Uncertainty in AI*, pages 339–347, 1991.

[36] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34 (6):871–882, 1986.

[37] Solomon E. Shimony. Relevant explanations: Allowing disjunctive assignments. In *Proceedings of the 9th Conference on Uncertainty in AI*, 1993.

[38] Solomon E. Shimony. The role of relevance in explanation I: Irrelevance as statistical independence. *International Journal of Approximate Reasoning*, 8(4):281–324, June 1993.

[39] Sclomon E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence Journal*, 68:399–410, 1994.

[40] Solomon E. Shimony and Eugene Charniak. A new algorithm for finding MAP assignments to belief networks. In *Proceedings of the 6th Conference on Uncertainty in AI*, pages 98–103, 1990.

[41] Michael P. Wellman and Chao-Lin Liu. State-space abstraction for anytime evaluation of probabilistic networks. In *Uncertainty in AI, Proceedings of the Tenth Conference*, pages 567–574, July 1994.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1994	3. REPORT TYPE AND DATES COVERED Technical Report

4. TITLE AND SUBTITLE
EXPLOITING IB ASSIGNMENTS FOR APPROXIMATING MARGINAL PROBABILITIES

5. FUNDING NUMBERS

6. AUTHOR(S)
Eugene Santos Jr. and Solomon Eyal Shimony

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/EN/TR94-06

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Computing marginal probabilities (whether prior or posterior) in Bayesian belief networks is a hard problem. This paper discusses deterministic approximation schemes that work by adding up the probability mass in a small number of value assignments to the network variables. Under certain assumptions, the probability mass in the union of these assignments is sufficient to obtain a good approximation. Such methods are especially useful for highly-connected networks, where the maximum clique size or the cutset size make the standard algorithms intractable.

In considering assignments, it is not necessary to assign values to variables that are independent of (d-separated from) the evidence and query nodes. In many cases, however, there is a finer independence structure not evident from the topology, but dependent on the conditional distributions of the nodes. We note that independence-based (IB) assignments, which were originally proposed as theory of abductive explanations, take advantage of such independence, and thus contain fewer assigned variables.

As a result, the probability mass in each assignment is greater than in the respective complete assignment. Thus, fewer IB assignments are sufficient, and a good approximation can be obtained more efficiently. IB assignments can be used for efficiently approximating posterior node probabilities even in cases which do not obey the rather strict skewness assumptions used in previous research. Two algorithms for finding the high probability IB assignments are suggested: one by doing a best-first heuristic search, and another by special-purpose integer linear programming. Experimental results show that this approach is feasible for highly connected belief networks.

14. SUBJECT TERMS
probabilistic reasoning, bayesian belief networks, relevance, belief updating, belief revision.

15. NUMBER OF PAGES
28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL